- ***-> reorganising arrays***
  - -> he has defined an example array
  - -> and then printed it out
  - -> by reorganising, he means reshaping
  - -> array_name.reshape(rows, columns)
  - -> this can be passed in as a 2x2
  - -> when you get errors, there can be a mismatch between the shape you're trying to put something into and the shape it currently is
  - -> dimensions are important when vertically stacking matrices

```
In [149]: before = np.array([[1,2,3,4],[5,6,7,8]])
          print(before)

          after = before.reshape((2,2,2))
          print(after)

          [[1 2 3 4]
           [5 6 7 8]]
          [[[1 2]
            [3 4]]

           [[5 6]
            [7 8]]]
```

- ***-> vector stacking***
  - -> you can take n vectors and stack them into one matrix
  - -> np.vstack([v1,v2]) <- these are now a part of the same matrix
  - -> and then in the arguments for it, you list out the different vectors which you want
  - -> the arguments you are passing into this methods are the names of the variables which store entire vectors
  - -> another example of this is np.hstack([v1,v2]) <- you stack them vertically

```
In [158]: # Vertically stacking vectors
          v1 = np.array([1,2,3,4])
          v2 = np.array([5,6,7,8])

          np.vstack([v1,v2,v1,v2])

Out[158]: array([[1, 2, 3, 4],
                 [5, 6, 7, 8],
                 [1, 2, 3, 4],
                 [5, 6, 7, 8]])
```

```
In [163]: # Horizontal  stack
          h1 = np.ones((2,4))
          h2 = np.zeros((2,2))

          np.hstack((h1,h2))

Out[163]: array([[1., 1., 1., 1., 0., 0.],
                 [1., 1., 1., 1., 0., 0.]])
```

- ***-> question***

What code would produce the following array?

[[1. 1.]
[1. 1.]
[1. 1.]
[1. 1.]]

a = np.ones((2, 4)) <- This one
b = a.reshape((4, 2))
print(b)

a = np.ones((2, 4))
b = a.reshape((2, 4))
print(b)

a = np.ones((2, 4))
b = a.reshape((8, 1))
print(b)